Effort Prediction Report

This document explains the rationale behind *Effort Prediction Reports,* a easy system to estimate projects at a early stage, i.e. when one does not know much about them.
*Effort Prediction Repor*ts are published under a *Creative Commons Attribution License* here: goo.gl/gpwbc

Methodological approach

Count

Break down the project to as many items as you can imagine
Estimating a future event is by any measure a difficult enterprise. If we have observed the event many times in the past, we can predict its sizes also in the future. Are projects like this?
If you have experienced the same before, lucky you. Mine are always different.

Calculate

Derive your estimate from data by building a quantitative model
Since I can not base my prediction on the last 10 text editors I wrote for a client that happens to request a text editor of the same kind, I need to rely on derived data. Data, I calculate from the records of my past projects. Hence, I only have to consult my archive of well documented project histories to extract the effort for similar tasks already done. Wait, you do not have such a archive, yet? Boy, you are doomed.

Estimate

Estimate the worst and the best cases
Well, as a last resort you can always guess. And I can help you in doing it better than you have done before. The quality of observation is fundamental for the process of prediction. Garbage in, Garbage out is the logic behind the accuracy and precision of any estimation. When you don't have much, you must be minute with what you got.

Eliciting subjective probabilities

First all tasks of a project are listed, the effort of each is calculated from indicators, if not possible, the task is guesstimated. Measure is the *Ideal Person Day (IPD*), this calibrates the eight-hour daily output of a qualified developer or analyst without distractions and interruptions.

The first estimate is for the *Worst Case*, this is held by the expert taking into account all the possible problems connected with the feature.

The second estimate is the *Best Case,*this is held by the expert taking into account shortcuts and serendipity the feature could bear, e.g. in software development: code reuse, framework deployment, free software.

Worst and Best Case give an estimate of the 90% confidence interval

With appropriate Calibrated Probability Assessment training a estimator reaches 90% accuracy that the real value is meet, in choosing the $[Worst\ case,\ Best\ case]$ range.

The confidence interval can be seen as the probability that the real effort is within the worst case - best case range 9 times for 10 features.

ID	Best case	Worst case	Realized Value
Feature 01	2	6	2
Feature 02	1	4	3
Feature 03	2	5	2
Feature 04	3	3	3
Feature 05	2	4	**5!!**
Feature 06	3	7	3
Feature 07	2	4	4
Feature 08	5	7	4
Feature 09	3	6	6
Feature 10	1.5	3	2

Illustration 1: The error of a feature estimation should be 1 in 10. Why not 100%? If we want perfect prediction under uncertainty the interval between Best Case and Worst Case would be so large that the value of information of such a prediction approaches zero. With 90% confidence we can predict an implementation effort under non-extreme situations.

The third estimate is the *Likely Case*: in the face of Best and Worst Case, what is the most Likely ?

A quantitative model of uncertainty

From Worst and best case, we calculate the *Expected Case* though a heuristic of *Program Evaluation, and Review Technique (PERT):*

$$Expected\ case\ =\ [Best\ case\ +\ (3\ *\ Likely\ case)\ +\ (2\ *\ Worst\ case)]\ /\ 6$$

The formula includes correction of the tendency for the optimistic estimate, see also http://portal.acm.org/citation.cfm?id=1408036

The sum of each case is the overall development effort. Precision errors in each feature point are smoothed by the summation. The effort in the range $A\ =\ [Best\ case,\ Worst\ case]$; $p(A)\ =\ 0.90$ gives us the variance and the aggregated standard deviation.

With this and the summed expected case as *expected value*, we can estimate a probability distribution of project effort assuming a Normal Distribution

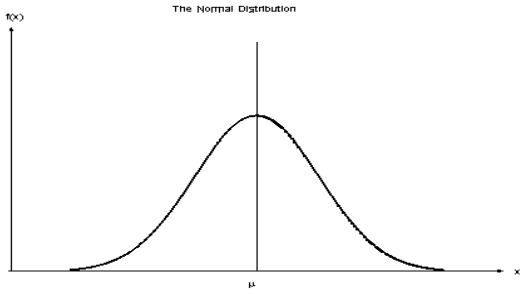

Illustration 2: The normal distribution with expected value μ

The Expected Case *μ* is the mean value and the standard deviaton *SD* defines the dispersion of the distribution. If we choose *μ* as our estimanted value we have a 50% chance of being late. so we choose a value bigger than *μ.* For example with a predicted effort of *μ + 2SD* we have a 98% probability of not underestimating the effort.

Multiplied by a market-conforming hourly budgeting total cost results in the forecast in €.

Critical analysis of the Normal Distribution assumption

The normal distribution is used due to its frequent occurrence in physical phenomena, but also because of its particularly favorable algebraic properties. A careless application implies to underestimate risk. In *project cost estimate* it is improbable to finish early but *likely to finish late*, this means the right leg of the distribution must be flat. Projects take longer than planned. A beta, triangular, binomial or Poisson distribution would be a better approach.

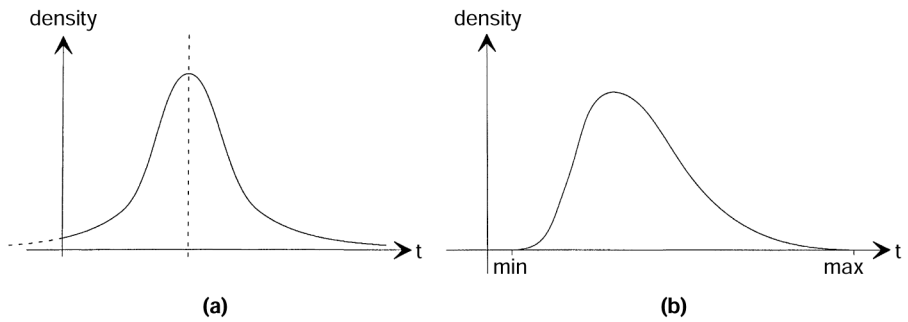

Illustration 3: The beta distribution (b) or with a flat right leg of the project cost rather than the Gaussian distribution curve (a)

The effect of this asymmetry is that the expected value of the distribution is no longer the same as the maximum is. Actually shifts the expected value of the maximum to right and is located on the flat leg of the distribution.

Interpretation of the systematic error by the Gaussian approximation

Many projects are either ahead of schedule or have an increased risk to take a long as planned. The confidence interval is systematically 'too optimistic'.
This is a vulnerability in the here used model, but one can expect that the estimation goodness is far superior to the widespread intuitive "guess-stimations". The very good algebraic properties of the Gauss function compensate somehow.